PR #24972 完整报告

sgl-project/sglang

[UnifiedTree]: Fix Unified HiCache tombstone lock release replay

合并时间：2026-05-12 13:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24972

执行摘要

- 一句话：修复 HiCache 临时锁错误释放恢复后的 tombstone 锁
- 推荐动作：值得精读。展示了如何通过记录操作历史来解决非幂等操作在多阶段并发中的正确性问题，设计模式具有参考价值。

功能与动机

PR body 明确指出：Record component node ids skipped during lock acquire and pass them back to release. This prevents a temporary admission lock from decrementing SWA or Mamba locks that were acquired later after HiCache load-back revived a tombstoned device value. 即避免临时准入锁在 HiCache load-back 恢复 tombstone 后错误释放后续获取的锁。

实现拆解

1. 扩展锁数据结构：在 IncLockRefResult 和 DecLockRefParams 中新增 skip_lock_node_ids: dict[ComponentType, set[int]] 字段，并在 IncLockRefResult.to_dec_params() 中传递该信息。
2. 修改调度器锁管理：schedule_policy.py 中的 _lock_node 方法直接调用 result.to_dec_params() 获取完整释放参数，移除旧的单独传递 swa_uuid_for_lock 方式，保证释放时携带所有需要跳过的节点 ID。
3. 在组件锁获取时记录跳过的节点：SWA、Mamba、Full 三个组件各自的 acquire_component_lock 在遇到 value 为 None（tombstone）时，将当前节点 ID 加入 result.skip_lock_node_ids。
4. 在组件锁释放时跳过这些节点：各组件的 release_component_lock 检查 params.skip_lock_node_ids，若当前节点 ID 在其中则直接返回，不执行 lock_ref 递减。
5. 新增单元测试：test_unified_radix_cache_unittest.py 增加两个测试方法，分别覆盖 SWA 和 Mamba 场景下临时锁、load-back 锁、请求锁的正确引用计数变化。

关键文件：

- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 缓存测试；类别 test；类型 test-coverage；符号 test_hicache_swa_temp_lock_does_not_release_restored_tombstone, test_hicache_mamba_temp_lock_does_not_release_restored_tombstone）：新增两个单元测试，分别验证 SWA 和 Mamba 场景下临时锁释放后不会错误减少

load-back 锁和请求锁，是 bugfix 的关键验证。

- python/sglang/srt/managers/schedule_policy.py（模块 调度策略；类别 source；类型 dependency-wiring；符号 _lock_node）：调度器的 _lock_node 是锁获取 / 释放的调用入口，改为统一使用 to_dec_params() 传递完整参数，移除了旧的条件分支，是修复的关键调用端。
- python/sglang/srt/mem_cache/base_prefix_cache.py（模块 缓存层；类别 source；类型 dependency-wiring；符号 IncLockRefResult, DecLockRefParams, IncLockRefResult.to_dec_params）：定义了 IncLockRefResult 和 DecLockRefParams 两个关键数据结构，新增 skip_lock_node_ids 字段和 to_dec_params 的对应传递逻辑，是整个修复的数据基础。
- python/sglang/srt/mem_cache/unified_cache_components/mamba_component.py（模块 缓存组件；类别 source；类型 core-logic；符号 acquire_component_lock, release_component_lock）：Mamba 组件的 acquire_component_lock 和 release_component_lock 是核心逻辑修改：在获取时记录跳过节点，释放时根据 skip 集合决定是否操作。
- python/sglang/srt/mem_cache/unified_cache_components/swa_component.py（模块 缓存组件；类别 source；类型 core-logic；符号 acquire_component_lock, release_component_lock）：SWA 组件的锁获取与释放逻辑同样需要修改，以支持 skip_lock_node_ids，修复与 Mamba 相同的 bug。
- python/sglang/srt/mem_cache/unified_cache_components/full_component.py（模块 缓存组件；类别 source；类型 core-logic；符号 acquire_component_lock）：Full 组件的 acquire_component_lock 有微小调整：原本构造新的 IncLockRefResult 实例并返回，现改为直接修改传入的 result 对象，以保留其他字段（如 skip_lock_node_ids）。

关键符号：IncLockRefResult.to_dec_params, SchedulePolicy._lock_node, MambaComponent.acquire_component_lock, MambaComponent.release_component_lock, SWAComponent.acquire_component_lock, SWAComponent.release_component_lock, FullComponent.acquire_component_lock, test_hicache_swa_temp_lock_does_not_release_restored_tombstone, test_hicache_mamba_temp_lock_does_not_release_restored_tombstone

关键源码片段

python/sglang/srt/mem_cache/base_prefix_cache.py

定义了 IncLockRefResult 和 DecLockRefParams 两个关键数据结构，新增 skip_lock_node_ids 字段和 to_dec_params 的对应传递逻辑，是整个修复的数据基础。

```
@dataclasses.dataclass
class IncLockRefResult:
    """Result of an inc_lock_ref operation."""
    delta: Optional[int] = None
    swa_uuid_for_lock: Optional[int] = None
    # Component nodes that were tombstones at acquire time. Replaying this set
    # at release prevents a short-lived lock from consuming a later load-back or
    # request lock after that tombstone becomes a valid device value.
```

```
    # 记录获取锁时因 tombstone 跳过的节点 ID，释放时跳过它们
    skip_lock_node_ids: dict[ComponentType, set[int]] = dataclasses.field(
        default_factory=dict
    )

    def to_dec_params(self) -> "DecLockRefParams":
        """Convert to the corresponding DecLockRefParams for dec_lock_ref."""
        return DecLockRefParams(
            swa_uuid_for_lock=self.swa_uuid_for_lock,
            skip_lock_node_ids={
                component_type: set(node_ids)
                for component_type, node_ids in self.skip_lock_node_ids.items()
            },
        )


@dataclasses.dataclass
class DecLockRefParams:
    """Parameters for dec_lock_ref operation."""
    swa_uuid_for_lock: Optional[int] = None
    # 释放时需跳过的节点 ID 集合，与 IncLockRefResult.skip_lock_node_ids 对应
    skip_lock_node_ids: dict[ComponentType, set[int]] = dataclasses.field(
        default_factory=dict
    )
```

python/sglang/srt/mem_cache/unified_cache_components/mamba_component.py

Mamba 组件的 acquire_component_lock 和 release_component_lock 是核心逻辑修改：在获取时记录跳过节点，释放时根据 skip 集合决定是否操作。

```
def acquire_component_lock(
    self, node: UnifiedTreeNode, result: IncLockRefResult
) -> IncLockRefResult:
    ct = self.component_type
    cd = node.component_data[ct]
    value = cd.value
    # 如果 value 为 None（tombstone），则记录节点 ID 到 result 中并返回，不增加 lock_ref
    if value is None:
        result.skip_lock_node_ids.setdefault(ct, set()).add(node.id)
        return result

    if cd.lock_ref == 0:
        vlen = len(value)
        self.cache.component_evictable_size_[ct] -= vlen
        self.cache.component_protected_size_[ct] += vlen
    cd.lock_ref += 1
    return result

def release_component_lock(
```

```
    self, node: UnifiedTreeNode, params: Optional[DecLockRefParams]
) -> None:
    ct = self.component_type
    cd = node.component_data[ct]
    # 获取需要跳过的节点 ID 集合
    skip_lock_node_ids = params.skip_lock_node_ids.get(ct, ()) if params else ()
    # 如果当前节点在跳过的集合中，则直接返回，不执行释放操作
    if node.id in skip_lock_node_ids:
        return

    value = cd.value
    if value is not None and cd.lock_ref > 0:
        if cd.lock_ref == 1:
            vlen = len(value)
            self.cache.component_evictable_size_[ct] += vlen
            self.cache.component_protected_size_[ct] -= vlen
        cd.lock_ref -= 1
```

评论区精华

review 中 ispobock 提问是否也需要修复旧的 SWA 或 Mamba 纯 radix cache（非 tree cache）。hzh0425 回答：这个方案是通用的，对于旧 cache，skip node 字段为空不会产生副作用，SWA UUID 仍然会传递。

- 是否需要兼容旧的 SWA/Mamba 纯 radix cache（非 tree cache）的锁行为？ (design): 确认改动向后兼容，无需单独处理旧 cache。

风险与影响

- 风险：主要风险在于 skip_lock_node_ids 在调用链中必须正确传递，若任何环节遗漏调用 to_dec_params() 或错误构造 DecLockRefParams，可能导致锁释放不足或过度释放。但本 PR 已在 _lock_node 统一使用 to_dec_params()，并在各组件的 release 中检查参数，风险较低。此外，base_prefix_cache.py 新增的 TYPE_CHECKING 导入可能引发循环依赖，但仅为类型检查时加载，无运行期风险。
- 影响：影响范围限定在启用 Unified Tree Cache 且使用 HiCache 的混合模型（如含 SWA 或 Mamba 的模型），修复了特定竞态条件下锁引用计数不一致的 bug，提升推理稳定性。对系统性能无显著影响。
- 风险标记：锁操作幂等性依赖，新字段传递正确性，测试覆盖有限（仅 SWA/Mamba）

关联脉络

- 暂无明显关联 PR